Universität Bremen
FB 3 – Informatik
Prof. Dr. Rainer Koschke
TutorIn: Sabrina Wilske

Software–Projekt 1 2013

VAK 03-BA-901.02

Architekturbeschreibung

IT_R3V0LUT10N

Sebastian Bredehöft	sbrede@tzi.de	2751589
Patrick Damrow	damsen@tzi.de	2056170
Tobias Dellert	tode@tzi.de	2936941
Tim Ellhoff	tellhoff@tzi.de	2520913
Daniel Pupat	dpupat@tzi.de	2703053

Abgabe: 22. Dezember 2013 — Version 1.1

Inhaltsverzeichnis

1 Einführung **3**
1.1 Zweck . . . 3
1.2 Status . . . 3
1.3 Definitionen, Akronyme und Abkürzungen . . . 3
1.4 Referenzen . . . 3
1.5 Übersicht über das Dokument . . . 3

2 Globale Analyse **3**
2.1 Einflussfaktoren . . . 3
2.1.1 Organisatorische Faktoren . . . 4
2.1.2 Technische Faktoren . . . 5
2.1.3 Produkt Faktoren . . . 7
2.2 Probleme und Strategien . . . 8

3 Konzeptionelle Sicht **9**

4 Modulsicht **9**

5 Datensicht **10**

6 Ausführungssicht **10**

7 Zusammenhänge zwischen Anwendungsfällen und Architektur **10**

8 Evolution **10**

Version und Änderungsgeschichte

Die aktuelle Versionsnummer des Dokumentes sollte eindeutig und gut zu identifizieren sein, hier und optimalerweise auf dem Titelblatt.

Version	Datum	Änderungen
1.0	25.11.2013	Dokumentvorlage als initiale Fassung kopiert
1.1	08.12.2013	Einflussfaktoren

1 Einführung

1.1 Zweck

Muss in SWP-2 ausgefüllt werden

Was ist der Zweck dieser Architekturbeschreibung? Wer sind die LeserInnen?

1.2 Status

Muss in SWP-2 ausgefüllt werden

1.3 Definitionen, Akronyme und Abkürzungen

1.4 Referenzen

1.5 Übersicht über das Dokument

Muss in SWP-2 ausgefüllt werden

2 Globale Analyse

2.1 Einflussfaktoren

Die Einflussfaktoren werden im Folgenden unterteilt in:

- Organisatorische Faktoren
- Technische Faktoren
- Produktfaktoren

2.1.1 Organisatorische Faktoren

Tabelle 1: Organisatorische Faktoren

O1	Time-To-Market
O2	Auslieferung von Produktfunktionen
O3	Budget
O4	Kenntnisse in Java und Android
O5	Kenntnisse in J-Unit
O6	Anzahl der Entwickler

O1	Time-To-Market
Faktor	Auslieferungsdatum 23.02.2014
Flexibilität und Veränderlichkeit	Die Deadline kann nicht verändert werden
Auswirkungen	Die Software muss zum Abgabedatum lauffähig sein

O2	Auslieferung von Produktfunktionen
Faktor	Alle Mindestanforderungen
Flexibilität und Veränderlichkeit	Es müssen alle Mindestanforderungen erfüllt sein, sind jedoch vom Kunden oder beim verlassen eines Gruppenmitglieds veränderbar
Auswirkungen	Architektur muss alle Mindestanforderungen abdecken, es muss darauf geachtet werden, dass diese sich im Verlauf noch ändern

O3	Budget
Faktor	Kein finanzielles Budget
Flexibilität und Veränderlichkeit	Es werden keine finanziellen Unterstützungen für das Produkt geben
Auswirkungen	Es können keine Kostenpflichtigen Dienste in Anspruch genommen werden

O4	Kenntnisse in Java und Android
Faktor	Kenntnisse der Entwickler in Java und Android
Flexibilität und Veränderlichkeit	Kenntnisse sind nicht flexibel, es muss in Java programmiert werden und über Smartphone laufen. Die Kenntnisse können sich im Laufe ändern, durch neue Erfahrungen und neu erworbenen Kenntnissen
Auswirkungen	Bei wenig Kenntnissen muss mehr Zeit eingeplant werden um sich diese anzueignen

O5	Kenntnisse in J-Unit
Faktor	Kenntnisse in J-Unit Tests
Flexibilität und Veränderlichkeit	Da Tests mit J-Unit gefordert werden, sind diese nicht verhandelbar oder Flexibel
Auswirkungen	Bei unzureichenden Tests kann es später beim Programm zu Problemen kommen, da Fehler spät oder gar nicht erkannt werden

O6	Anzahl der Entwickler
Faktor	Die Anzahl der Entwickler
Flexibilität und Veränderlichkeit	Es können keine neuen Gruppenmitglieder dazukommen, es können aber jederzeit Gruppenmitglieder wegfallen
Auswirkungen	Wenn Gruppenmitglieder wegfallen, müssen die restlichen mehr Arbeiten und mehr Zeit einplanen. Auch müssen Projektplan und Architektur neu angepasst werden.

2.1.2 Technische Faktoren

Tabelle 2: Technische Faktoren

T1	Software funktioniert unter Windows, Linux und MacOS
T2	Software funktioniert als App(Andriod 2.3 oder höher)
T3	SQL-Datenbank
T4	Mehrere parallele Nutzer
T5	Client-Server System
T6	Benutzerschnittstelle
T7	Implementierungssprache Java
T8	Beschränkungsfreiheit für Fremdbibliotheken

T1	Software funktioniert unter Windows, Linux und MacOS
Faktor	Die Software muss auf den Betriebssystemen Windows, Linux und MacOS laufen
Flexibilität und Veränderlichkeit	nicht Flexibel, da dies zu den Mindestanforderungen gehört. Veränderungen können jederzeit vom Kunden vorgenommen werden.
Auswirkungen	Die Entwickler müssen sich mit allen Betriebsprogrammen befassen und sichergehen, dass es auf allen funktioniert

T2	Software funktioniert als App (Andriod 2.3 oder höher)
Faktor	Die Software muss als Android App auf einem Smartphone laufen
Flexibilität und Veränderlichkeit	nicht Flexibel, da dies zu den Mindestanforderungen gehört. Veränderungen können jederzeit vom Kunden vorgenommen werden.
Auswirkungen	Die Software muss wie gefordert als App auf einem Android-Smartphone laufen

T3	SQL-Datenbank
Faktor	Software läuft über eine relationale Datenbank
Flexibilität und Veränderlichkeit	Flexibel jedoch muss eine Datenbank mit SQL oder SQL-ähnlichen abfragen verwendet werden
Auswirkungen	Es muss eine relationale Datenbank für die serverseitige Persistenz benutzt werden. Es muss eine Datenbank mit SQL oder SQL-ähnlichen abfragen verwendet werden

T4	Mehrere parallele Nutzer
Faktor	Es greifen mehrere Nutzer zur gleichen Zeit auf die Software zu
Flexibilität und Veränderlichkeit	Es ist uns überlassen, wie viele Nutzer zur gleichen Zeit auf das System zugreifen dürfen
Auswirkungen	Die Software muss darauf ausgelegt sein, mehrere Nutzer zur gleichen Zeit zu verwalten

T5	Client-Server System
Faktor	Die Software arbeitet über ein Client-Server System
Flexibilität und Veränderlichkeit	Da wir übers Internet auf den Server zugreifen müssen, ist es notwendig ein Server-Client System zu verwenden
Auswirkungen	Die Implementierung wird in Server und Client aufgeteilt(siehe 3) Übers Internet werden die Daten zwischen Server und Client ausgetauscht

T6	Benutzerschnittstelle
Faktor	
Flexibilität und Veränderlichkeit	
Auswirkungen	

T7	Implementierungssprache Java
Faktor	Die Software muss in Java 5 oder höher geschrieben werden
Flexibilität und Veränderlichkeit	Nicht Flexibel, da dies zu den Mindestanforderungen gehört
Auswirkungen	Die Software muss in Java geschrieben werden, daher müssen alle Entwickler diese Sprache beherrschen

T8	Beschränkungsfreiheit für Fremdbibliotheken
Faktor	Fremdbibliotheken dürfen für den Einsatz in Forschung und Lehre keine Beschränkungen aufweisen
Flexibilität und Veränderlichkeit	Nicht Flexibel, da dies zu den Mindestanforderungen gehört
Auswirkungen	Es darf keine Software oder Bibliothek verwendet werden, die Kostenpflichtig ist

2.1.3 Produkt Faktoren

Tabelle 3: Produkt Faktoren

P1	Mindestanforderung
P2	Performanz
P3	Benutzerrechte
P4	Fehlererkennung

P1	Mindestanforderung
Faktor	Das Produkt muss alle Mindestanforderungen enthalten
Flexibilität und Veränderlichkeit	Alle Anforderungen müssen zum Bestehen erfüllt werden. Die Anforderungen können vom Kunden oder Dozenten verändert werden oder die Anforderungen werden bei einem Austritt eines Mitglieds verringert.
Auswirkungen	Es müssen alle Mindestanforderungen implementiert werden

P2	Performanz
Faktor	Möglichst schnelle Ausführungszeiten
Flexibilität und Veränderlichkeit	Flexibel, da nichts davon in den Mindestanforderungen steht
Auswirkungen	Es sollte bei der Implementierung auf einen schnellen Datenaustausch zwischen Server und Client geachtet werden

P3	Benutzerrechte
Faktor	Es gibt verschiedene Benutzer mit unterschiedlichen Rechten
Flexibilität und Veränderlichkeit	Nicht Flexibel, da dies vom Kunden gefordert wird
Auswirkungen	Es müssen unterschiedliche Benutzer implementiert werden, die unterschiedliche Rechte haben und diese auch nicht überschreiten dürfen

P4	Fehlererkennung
Faktor	Fehler sollten von der Software erkannt werden und entsprechend behandelt werden
Flexibilität und Veränderlichkeit	Flexibel, da dies nicht ausdrücklich vom Kunden gefordert wird
Auswirkungen	Fehler müssen erkannt werden und durch Exception muss es dann entsprechend Korrigiert werden. Die Software sollte weiter laufen

2.2 Probleme und Strategien

Aus einer Menge von Faktoren ergeben sich Probleme, die nun in Form von Problemkarten beschrieben werden. Diese resultieren z.B. aus

- *Grenzen oder Einschränkungen durch Faktoren*
- *der Notwendigkeit, die Auswirkung eines Faktors zu begrenzen*
- *der Schwierigkeit, einen Produktfaktor zu erfüllen, oder*
- *der Notwendigkeit einer allgemeinen Lösung zu globalen Anforderungen.*

Dazu entwickelt Ihr Strategien, um mit den identifizierten Problemen umzugehen.

Achtet auch hier darauf, dass die Probleme und Strategien wirklich die Architektur betreffen und nicht etwa das Projektmanagement. Die Strategien stellen im Prinzip die Designentscheidungen dar. Sie sollten also die Erklärung für den konkreten Aufbau der verschiedenen Sichten liefern.

Beschreibt möglichst mehrere Alternativen und gebt an, für welche Ihr Euch letztlich aus welchem Grunde entschieden habt. Natürlich müssen die genannten Strategien in den folgenden Sichten auch tatsächlich umgesetzt werden!

Ein sehr häufiger Fehler ist es, dass SWP-Gruppen arbeitsteilig vorgehen: die eine Gruppe schreibt das Kapitel zur Analyse von Faktoren und zu den Strategien, die andere Gruppe beschreibt die diversen Sichten, ohne dass diese beiden Gruppen sich abstimmen. Natürlich besteht aber ein Zusammenhang zwischen den Faktoren, Strategien und Sichten. Dieser muss erkennbar sein, indem sich die verschiedenen Kapitel eindeutig aufeinander beziehen.

3 Konzeptionelle Sicht

Diese Sicht beschreibt das System auf einer hohen Abstraktionsebene, d.h. mit sehr starkem Bezug zur Anwendungsdomäne und den geforderten Produktfunktionen und -attributen. Sie legt die Grobstruktur fest, ohne gleich in die Details von spezifischen Technologien abzugleiten. Sie wird in den nachfolgenden Sichten konkretisiert und verfeinert. Die konzeptionelle Sicht wird mit UML-Komponentendiagrammen visualisiert.

4 Modulsicht

Diese Sicht beschreibt den statischen Aufbau des Systems mit Hilfe von Modulen, Subsystemen, Schichten und Schnittstellen. Diese Sicht ist hierarchisch, d.h. Module werden in Teilmodule zerlegt. Die Zerlegung endet bei Modulen, die ein klar umrissenes Arbeitspaket für eine Person darstellen und in einer Kalenderwoche implementiert werden können. Die Modulbeschreibung der Blätter dieser Hierarchie muss genau genug und ausreichend sein, um das Modul implementieren zu können.

Die Modulsicht wird durch UML-Paket- und Klassendiagramme visualisiert.

Die Module werden durch ihre Schnittstellen beschrieben. Die Schnittstelle eines Moduls M ist die Menge aller Annahmen, die andere Module über M machen dürfen, bzw. jene Annahmen, die M über seine verwendeten Module macht (bzw. seine Umgebung, wozu auch Speicher, Laufzeit etc. gehören). Konkrete Implementierungen dieser Schnittstellen sind das Geheimnis des Moduls und können vom Programmierer festgelegt werden. Sie sollen hier dementsprechend nicht beschrieben werden.

Die Diagramme der Modulsicht sollten die zur Schnittstelle gehörenden Methoden enthalten. Die Beschreibung der einzelnen Methoden (im Sinne der Schnittstellenbeschreibung) geschieht allerdings per Javadoc im zugehörigen Quelltext. Das bedeutet, dass Ihr für alle Eure Module Klassen, Interfaces und Pakete erstellt und sie mit den Methoden der Schnittstellen verseht. Natürlich noch ohne Methodenrümpfe bzw. mit minimalen Rümpfen. Dieses Vorgehen vereinfacht den Schnittstellenentwurf und stellt Konsistenz sicher.

Jeder Schnittstelle liegt ein Protokoll zugrunde. Das Protokoll beschreibt die Vor- und Nachbedingungen der Schnittstellenelemente. Dazu gehören die erlaubten Reihenfolgen, in denen Methoden der Schnittstelle aufgerufen werden dürfen, sowie Annahmen über Eingabeparameter und Zusicherungen über Ausgabeparameter. Das Protokoll von Modulen wird in der Modulsicht beschrieben. Dort, wo es sinnvoll ist, sollte es mit Hilfe von Zustands- oder Sequenzdiagrammen spezifiziert werden. Diese sind dann einzusetzen, wenn der Text allein kein ausreichendes Verständnis vermittelt (insbesondere bei komplexen oder nicht offensichtlichen Zusammenhängen).

Der Bezug zur konzeptionellen Sicht muss klar ersichtlich sein. Im Zweifel sollte er explizit erklärt werden. Auch für diese Sicht muss die Entstehung anhand der Strategien erläutert werden.

5 Datensicht

Hier wird das der Anwendung zugrundeliegende Datenmodell beschrieben. Hierzu werden neben einem erläuternden Text auch ein oder mehrere UML-Klassendiagramme verwendet. Das hier beschriebene Datenmodell wird u.a. jenes der Anforderungsspezifikation enthalten, allerdings mit implementierungsspezifischen Änderungen und Erweiterungen. Siehe die gesonderten Hinweise.

6 Ausführungssicht

Muss in SWP-2 ausgefüllt werden

Die Ausführungssicht beschreibt das Laufzeitverhalten. Hier werden die Laufzeitelemente aufgeführt und beschrieben, welche Module sie zur Ausführung bringen. Ein Modul kann von mehreren Laufzeitelementen zur Laufzeit verwendet werden. Die Ausführungssicht beschreibt darüber hinaus, welche Laufzeitelemente spezifisch miteinander kommunizieren. Zudem wird bei verteilten Systemen (z.B. Client-Server-Systeme) dargestellt, welche Module von welchen Prozessen auf welchen Rechnern ausgeführt werden.

7 Zusammenhänge zwischen Anwendungsfällen und Architektur

In diesem Abschnitt sollen Sequenzdiagramme mit Beschreibung(!) für zwei bis drei von Euch ausgewählte Anwendungsfälle erstellt werden. Ein Sequenzdiagramm beschreibt den Nachrichtenverkehr zwischen allen Modulen, die an der Realisierung des Anwendungsfalles beteiligt sind. Wählt die Anwendungsfälle so, dass nach Möglichkeit alle Module Eures entworfenen Systems in mindestens einem Sequenzdiagramm vorkommen. Falls Euch das nicht gelingt, versucht möglichst viele und die wichtigsten Module abzudecken.

8 Evolution

Muss in SWP-2 ausgefüllt werden

Beschreibt in diesem Abschnitt, welche Änderungen Ihr vornehmen müsst, wenn sich Anforderungen oder Rahmenbedingungen ändern. Insbesondere sollten hierbei die in der Anforderungsspezifikation unter „Ausblick“ bereits genannten Punkte behandelt werden.

...